



Automation of your work process

What is Phing?

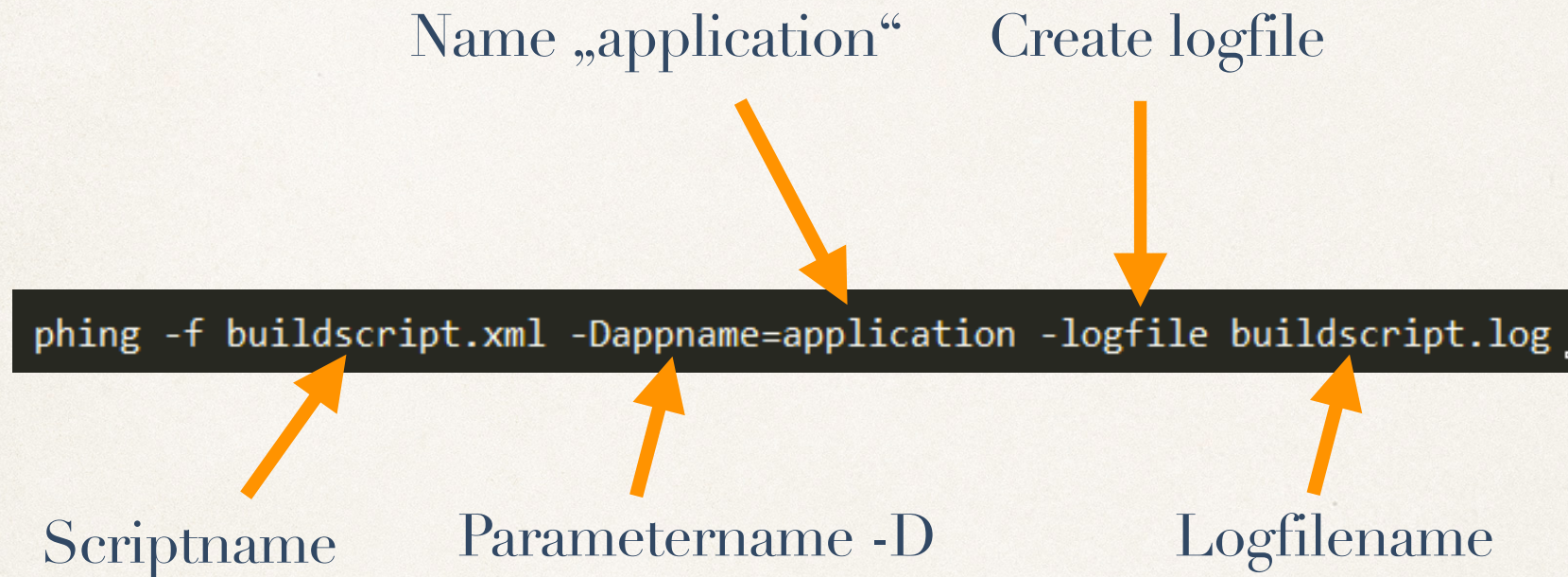
XML-based tool for task automation.

What can you do with Phing?

- ❖ Create, copy, move files and folders
- ❖ Running apps/tools (exp. Editor)
- ❖ Checking syntax, codeguidelines
- ❖ Running Tests
- ❖ Running Shell-Scripts, PHP-Scripts
- ❖ Detect code quality
- ❖ Manage repositories
- ❖ Send notifications
- ❖ Convert datas
- ❖ Server deployment

Basics

Run Buildscript



Structure of a build script

Start with this task

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="teiresias" default="main" basedir=".">

  <target name="main">
    <phingcall target="-task1" />
    <phingcall target="-task2" />
  </target>

  <target name="-task0">
    <echo message="Start Task 0"/>
  </target>

  <target name="-task1" depends="-task0">
    <echo message="Start Task 1"/>
  </target>

  <target name="-task2" depends="-task0">
    <echo message="Start Task 2"/>
  </target>

</project>
```

Call sub-task

TARGET
Task definition

Run this task before

Variables and Parameters

Parameter

-Dappname=application

```
<project name="teiresias" default="main" basedir=".">

  <!-- Name der App (wird als Parameter -appname beim Aufruf übergeben -->

    <property name="app_name" value="${appname}" />

  <!-- Pfade definieren -->

    <property name="app_path" value="app" />
    <property name="common_path" value="../common/app" />
    <property name="app_build_path" value="../../../build/${app_name}" />
    <property name="build_path" value="${app_build_path}/app" />
    <property name="apidoc_path" value="${app_build_path}/apidoc" />

</project>
```

The diagram illustrates the relationship between a parameter and variables in a build file. An orange arrow points from the parameter `-Dappname=application` to the `value="${appname}"` in the `<property name="app_name" />` line. Another orange arrow points from the `<property name="app_name" />` line to the `value="app"` in the `<property name="app_path" />` line. A third orange arrow points from the `value="${app_name}"` in the `<property name="app_build_path" />` line to the `value="${app_build_path}/app"` in the `<property name="build_path" />` line. A fourth orange arrow points from the `value="${app_build_path}/apidoc"` in the `<property name="apidoc_path" />` line to the `value="${app_build_path}/apidoc"` in the `<property name="apidoc_path" />` line.

Define variables: PROPERTY

Call a variable `${...}`

Group files: FILESET

```
<!-- Auswahl der Dateien, die geprüft werden sollen -->
```

```
<!-- Auswahl aller Dateien der App -->
```

```
<fileset id="app_project_files" dir="${app_path}">
```

```
<include name="*.*" />
```

```
<include name="**/*.*)" />
```

```
</fileset>
```

Path

All files

```
<!-- Auswahl aller gemeinsamen Dateien -->
```

```
<fileset id="common_project_files" dir="${common_path}">
```

```
<include name="*.*" />
```

```
<include name="**/*.*)" />
```

```
</fileset>
```

All subfolders

ID for referencing to the file set

Installation

Installation with Composer

composer.json

```
{  
  "require" : {  
    "squizlab/php_codesniffer": "dev-phpcs-fixer",  
    "phpunit/phpunit" : "3.8.*@dev",  
    "phpdocumentator/phpdocumentator" : "2.*",  
    "phing/phing": "dev-master"  
  }  
}
```

Run in Shell `composer install`

Table of Content

Content

1. Select files to commit
2. Check filename
3. Check syntax
4. Check coding guideline
5. Run Unit-Tests
6. Create documentation
7. Check you files into your repository

Start to commit your files



git pre commit hook

```
echo "Run Build Process:"  
php -d html_errors=off -qC /c/xampp/php/pear/phing.php  
-f ../../buildtools/buildscripts/pre_commit_build.xml  
-logfile _log/pre_commit_build.log || exit 1
```

Start Phing



Logfile for results



Start pre_commit_build



Define a task list

```
<target name="main">
    <echo message="-----" />
    <echo message="| " />
    <echo message="| Pre_Commit_Build" />
    <echo message="| " />
    <echo message="-----" />

    <phingcall target="-filter_uncommitted_files_from_git" />
    <phingcall target="-list_collected_files" />
    <!-- <phingcall target="-check_folder_names" /> -->
    <phingcall target="-check_file_names" />
    <phingcall target="-check_lineend" />
    <phingcall target="-check_utf8" />
    <phingcall target="-check_syntax" />
    <phingcall target="-check_coding_guideline" />
    <phingcall target="-check_code_duplication" />
    <phingcall target="-unit_test" />
    <phingcall target="-generate_documentation" />

</target>
```

TARGET main

Call job

1: Select new files from git

Select files from git: TMP_FILE

```
<target name="-filter_uncommitted_files_from_git">

    <echo message="+=====+/">
    <echo message="|"/>
    <echo message="| Filter uncommitted files from GIT"/>
    <echo message="|"/>
    <echo message="+=====+/">

    <exec command="git diff-index --name-only --cached --diff-filter=ACMR HEAD" output="_tmp/uncommitted_files.txt" />

    <echo msg="Fix" />
</target>
```



Run Git command



Store filenames in text file

Select files from git: FILE TYPE

```
<target name="-collect_files">
```

```
  <echo msg="" />
```

```
  <!-- Auswahl aller PHP-Dateien -->
```

```
    <fileset id="php_project_files" dir="." includesfile="_tmp/uncommitted_files.txt">
      <filename name="**.*.php" />
      <exclude name="app/system/scripts/**/*.*.php" />
    </fileset>
```

```
  <!-- Auswahl aller HTML-Dateien -->
```

```
    <fileset id="html_project_files" dir="." includesfile="_tmp/uncommitted_files.txt">
      <filename name="**.*.html" />
    </fileset>
```

```
  <!-- Auswahl aller Javascript-Dateien -->
```

```
    <fileset id="javascript_project_files" dir="." includesfile="_tmp/uncommitted_files.txt">
      <filename name="**.*.js" />
      <exclude name="app/system/scripts/**/*.*.js" />
    </fileset>
```

Include PHP files

List of files

Exclude files in SCRIPT folder

List selected files

```
<target name="-list_collected_files" depends="-collect_files">

    <echo message="=====+</pre>"/>


```

 <echo message="|
 <echo message="| Collect Project Files for Build
 <echo message="|
 <echo message="=====+</pre>"/>


```

    <!-- Auswahl aller PHP-Dateien -->

    <echo msg="PHP-Files:" />

    <echo>
        <fileset refid="php_project_files" />
    </echo>

    <!-- Auswahl aller HTML-Dateien -->

    <echo msg="HTML-Files:" />

    <echo>
        <fileset refid="html_project_files" />
    </echo>
</target>

```


```


```

Run this job
before

Fileset ID

List files

2: Check filename

Check filenames

```
<target name="-check_file_names" depends="-collect_files">

    <echo message="=====+"/>
    <echo message="|"/>
    <echo message="| Check Filenames"/>
    <echo message="|"/>
    <echo message="=====+"/>

    <!-- Definition eines adhoc-task, der die Dateinamen prüft -->

    <!-- /**
    * Ad-Hoc-Task Check-Filesnames
    *
    * Definition eines adhoc-task, der die Dateinamen prüft
    *
    * @todo Den Ad-Hoc-Task in einen eigenen Task auslagern
    */ -->

    <!-- Variablen/Properties vordefinieren -->

    <property name="task_failed" value="0" />
    <property name="file_check_failed" value="0" />
    <property name="total_number_of_files" value="0" />
    <property name="failed_files" value="0" />


```


Run PHP in Adhoc-Task: Definitio

```
<!-- Definition des AdhocTasks -->

<adhoc-task name="check_filename"><![CDATA[
class CheckFileName extends Task {


    /** Any filesets of files that should be appended.
     *
     */

    private $filesets = array();

    /**
     * Nested creator, adds a set of files (nested <fileset> attribute).
     * This is for when you don't care what order files get appended.
     * @return FileSet
     */

    function createFileSet() {
        $num = array_push($this->filesets, new FileSet());
        return $this->filesets[$num-1];
    }

    /**
     *
     */
}
```



Run PHP in Adhoc-Task: Code

```
function main() {  
  
    // Variablen initiieren  
  
    $this->project->setProperty('file_check_failed', '0');  
    $total_number_of_files = 0;  
    $failed_files=0;  
  
    // append any files in filesets  
  
    foreach($this->filesets as $fs) {  
        try {  
            // ...  
        }  
  
        // Ergebnisse in Property speichern  
  
        $this->project->setProperty('total_number_of_files', $total_number_of_files  
);  
        $this->project->setProperty('failed_files', $failed_files);  
  
    } catch (BuildException $be) {  
        $this->log($be->getMessage(), Project::MSG_WARN);  
    }  
}  
}
```

Store value in a Property



```
]]></adhoc-task>
```


Run Adhoc-Task

My Adhoc-Task

```
<echo message="+-----+"/>
<echo message="|"/>
<echo message="| Check PHP Filenames"/>
<echo message="|"/>
<echo message="+-----+"/>

<check_filename>
  <fileset refid="php_project_files" />
</check_filename>

<echo message="Anzahl der getesteten Dateien: ${total_number_of_files}" />
<echo message="Anzahl der fehlerhaften Dateien: ${failed_files}" />
<echo message="Fehler ${file_check_failed}" />
```


Task failed?

```
<!-- /**
 * Build Failed?
 *
 * Prüfen, ob das Build fehlschlagen soll, da ein Fehler gefunden wurde.
 */ -->

<if>
  <equals arg1="${task_failed}" arg2="1" />
  <then>
    <fail message="Task failed" />
  </then>
</if>
```



Property form
Adhoc-Task

3: Check syntax

Check syntax

```
<target name="-check_syntax" depends="-collect_files">

    <echo message="=====+>"/>
    <echo message="|"/>
    <echo message="| Syntax-Check"/>
    <echo message="|"/>
    <echo message="=====+>"/>

    <!-- /**
    * Prüfung der PHP-Dateien
    *
    */ -->

    <echo message="-----+>"/>
    <echo message="|"/>
    <echo message="| PHP Syntax-Check"/>
    <echo message="|"/>
    <echo message="-----+>"/>

    <phplint level="verbose" haltonfailure="true">
        <fileset refid="php_project_files" />
    </phplint>

    <echo message="" />
</target>
```

PHPLint

STOP

4: Check coding guideline

Check coding guideline

```
<target name="-check_coding_guideline" depends="-collect_files">

    <echo message="=====+>"/>
    <echo message="|"/>
    <echo message="| Check Coding-Guideline"/>
    <echo message="|"/>
    <echo message="=====+>"/>

    <!-- /**
     * PHP-Code prüfen
     *
     * @require PHP_CodeSniffer
     */ -->

    <!-- Prüfen auf PSR-1 -->

    <echo message="-----+>"/>
    <echo message="|"/>
    <echo message="| PHP Check PSR-1 Coding-Guideline"/>
    <echo message="|"/>
    <echo message="-----+>"/>

    <phpcodesniffer standard="PSR1" haltonerror="false">
        <fileset refid="php_project_files" />
    </phpcodesniffer>
</target>
```

Define Code Guideline

CodeSniffer

5: Run unit-tests

Run Unit Tests

PHPUnitTest

```
<target name="-unit_test" depends="-collect_files">

    <echo message="=====+</>" />
    <echo message="|</>" />
    <echo message="| UnitTests|</>" />
    <echo message="|</>" />
    <echo message="=====+</>" />

    <!-- /**
     * PHP-Unit-Tests
     *
     * @require PHPUnit
     * @todo FileSet muß noch so angepaßt werden, dass die Tests gefunden werden.
     */ -->

    <echo message="-----+</>" />
    <echo message="|</>" />
    <echo message="| PHP-UnitTest|</>" />
    <echo message="|</>" />
    <echo message="-----+</>" />

    <phpunit haltonfailure="false" printsummary="true">

        <batchtest>
            <fileset dir="app">
                <include name="*_unittest.php"/>
                <include name="**/*_unittest.php"/>
            </fileset>
        </batchtest>

    </phpunit>

</target>
```


6: Create documentation

Create Documentation

```
<target name="-create_api_documentation">
    <echo message="=====+</>" />
    <echo message="|</>" />
    <echo message="| Generate API Documentation</>" />
    <echo message="|</>" />
    <echo message="=====+</>" />

    <phpdoc2 title="API Documentation" destdir="${apidoc_path}" template="responsive-twig">
        <fileset refid="php_project_files" />
    </phpdoc2>

    <echo message="" />
</target>
```

Use this file set

Destination folder

Design template

*** DONE ***

Predefined Tasks

B.1. AdhocTaskdefTask
B.2. AdhocTypedefTask
B.3. AppendTask
B.4. ApplyTask
B.5. AvailableTask
B.6. ChmodTask
B.7. ChownTask
B.8. ConditionTask
B.9. CopyTask
B.10. CvsTask
B.11. CvsPassTask
B.12. DeleteTask
B.13. EchoTask
B.14. ExecTask
B.15. FailTask
B.16. ForeachTask
B.17. IfTask
B.18. ImportTask
B.19. IncludePathTask
B.20. InputTask
B.21. LoadFileTask
B.22. MkdirTask
B.23. MoveTask
B.24. PhingTask
B.25. PhingCallTask
B.26. PhpEvalTask
B.27. PropertyTask
B.28. PropertyPromptTask
B.29. ReflexiveTask
B.30. ResolvePathTask

B.31. TaskdefTask
B.32. TouchTask
B.33. TryCatchTask
B.34. TstampTask
B.35. TypedefTask
B.36. UpToDateTask
B.37. WaitForTask
B.38. XsltTask

[C.1. ApiGenTask](#)
[C.2. CoverageMergerTask](#)
[C.3. CoverageReportTask](#)
[C.4. CoverageSetupTask](#)
[C.5. CoverageThresholdTask](#)
[C.6. DbDeployTask](#)
[C.7. DocBloxTask](#)
[C.8. ExportPropertiesTask](#)
[C.9. FileHashTask](#)
[C.10. FileSizeTask](#)
[C.11. FileSyncTask](#)
[C.12. FtpDeployTask](#)
[C.13. GitInitTask](#)
[C.14. GitCloneTask](#)
[C.15. GitGcTask](#)
[C.16. GitBranchTask](#)
[C.17. GitFetchTask](#)
[C.18. GitCheckoutTask](#)
[C.19. GitCommitTask](#)
[C.20. GitMergeTask](#)
[C.21. GitPullTask](#)
[C.22. GitPushTask](#)

[C.23. GitTagTask](#)
[C.24. GitLogTask](#)
[C.25. GrowlNotifyTask](#)
[C.26. HttpGetTask](#)
[C.27. HttpRequestTask](#)
[C.28. IoncubeEncoderTask](#)
[C.29. IoncubeLicenseTask](#)
[C.30. JslLintTask](#)
[C.31. JsMinTask](#)
[C.32. LiquibaseChangeLogTask](#)
[C.33. LiquibaseDbDocTask](#)
[C.34. LiquibaseDiffTask](#)
[C.35. LiquibaseRollbackTask](#)
[C.36. LiquibaseTagTask](#)
[C.37. LiquibaseUpdateTask](#)
[C.38. MailTask](#)
[C.39. ParallelTask](#)
[C.40. PatchTask](#)
[C.41. PDOSQLiteExecTask](#)
[C.42. PearPackageTask](#)
[C.43. PearPackage2Task](#)
[C.44. PharPackageTask](#)
[C.45. PhkPackageTask](#)
[C.46. PhpCodeSnifferTask](#)
[C.47. PHPCPDTask](#)
[C.48. PHPLocTask](#)
[C.49. PHPMDTask](#)
[C.50. PhpDependTask](#)
[C.51. PhpDocumentorTask](#)
[C.52. DocBloxTask](#)
[C.53. PhpDocumentorExternalTask](#)

[C.54. PhpLintTask](#)
[C.55. PHPUnitTask](#)
[C.56. PHPUnitReport](#)
[C.57. rSTTask](#)
[C.58. S3PutTask](#)
[C.59. S3GetTask](#)
[C.60. ScpTask](#)
[C.61. SshTask](#)
[C.62. SimpleTestTask](#)
[C.63. SvnCheckoutTask](#)
[C.64. SvnCommitTask](#)
[C.65. SvnCopyTask](#)
[C.66. SvnExportTask](#)
[C.67. SvnInfoTask](#)
[C.68. SvnLastRevisionTask](#)
[C.69. SvnListTask](#)
[C.70. SvnLogTask](#)
[C.71. SvnUpdateTask](#)
[C.72. SvnSwitchTask](#)
[C.73. SymfonyConsoleTask](#)
[C.74. SymlinkTask](#)
[C.75. TarTask](#)
[C.76. UntarTask](#)
[C.77. UnzipTask](#)
[C.78. VersionTask](#)
[C.79. WikiPublishTask](#)
[C.80. XmlLintTask](#)
[C.81. XmlPropertyTask](#)
[C.82. ZendCodeAnalyzerTask](#)
[C.83. ZendGuardEncodeTask](#)
[C.84. ZendGuardLicenseTask](#)
[C.85. ZipTask](#)

Links

✿ www.phing.info